SET-MM – A Software Evaluation Technology Maturity Model

Raúl García-Castro
Ontology Engineering Group
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Facultad de Informática, Universidad Politécnica de Madrid, Spain
rgarcia@fi.upm.es

***Abstract*—The application of software evaluation technologies in different research fields to verify and validate research is a key factor in the progressive evolution of those fields. Nowadays, however, to have a clear picture of the maturity of the technologies used in evaluations or to know which steps to follow in order to improve the maturity of such technologies is not easy. This paper describes a Software Evaluation Technology Maturity Model that can be used to assess software evaluation technologies in a research field. To illustrate the use of this model, we have employed it for assessing the maturity of software evaluation technologies in some evaluation initiatives within the semantic research field.**

I. Introduction

For research to reach maturity it is necessary to improve the quality of the research processes and their results over time. However, to apply the vague concept of maturity to a research field, which is itself difficult to define, may seem an arduous task.

It is well known that any research field has to periodically assess its status by measuring different aspects of it (e.g., outreach to other fields, quality of publications, technology maturity) and that the results obtained from these assessments are the ones that initiate the actions needed towards improvement and maturity in the field.

Nevertheless, to perform successful assessments for choosing the most appropriate actions, it is necessary to have clear goals and be aware of what is needed to achieve those goals.

Maturity models have been used for decades to guide improvement by providing both a framework of incremental maturity levels to be compared against and different measurable goals that must be satisfied to achieve each maturity level.

The most relevant maturity models are probably the Capability Maturity Model (CMM or SW-CMM) [1] and its successor, the Capability Maturity Model Integration[1] (CMMI). Both maturity models have been defined by the Software Engineering Institute with the goal of improving an organization's software development processes.

Other maturity models have also been defined, most of them inspired by CMM or CMMI, mainly in the software engineering domain but also in domains such as the organization management one (e.g., the Portfolio, Programme and Project Management Maturity Model [2]) or the e-learning one (the e-learning Maturity Model [3]).

One of the measurable aspects of a research field is what technologies support the evaluation of the research outcomes. In fact, the application of software evaluation technologies in any research field to verify and validate research is a key factor for research evolution by means of experimentation-driven research [4].

The goal of this paper is to describe a Software Evaluation Technology Maturity Model (SET-MM) that can be used to assess software evaluation technologies in any research field.

Similarly to other maturity models, the goal of the SET-MM is not to derive a figure for a specific level but to provide some guiding to improve the current activities in the research field.

To illustrate how the SET-MM maturity model functions, we have used it to assess the maturity of software evaluation technologies in some evaluation initiatives within a concrete research field, that of semantic research.

This paper is structured as follows. Section II introduces other maturity models that are related to the one here presented as they cover similar domains. Then, Section III presents the scope of this work and the process followed to define the maturity model, whereas Section IV enumerates the assumptions taken into account when defining this maturity model. Section V describes the five maturity levels of SET-MM and the main notions behind the model. Section VI presents how we have used the maturity model here presented to assess the maturity of software evaluation technologies in the semantic research field. Finally, Section VII draws the conclusions from this work and proposes future lines of research.

II. Related Work

This section presents other maturity models that cover similar domains and that have served us as input for defining the maturity model presented in this paper.

The *Capability Maturity Model for Software* was defined by the Software Engineering Institute in the early 1990s [1]. Since then different maturity models have appeared, each collecting the best practices to be used when comparing an organization's practices and guiding process improvement.

The proliferation of such models has led to their combination into a single improvement framework, namely, the *Capability Maturity Model Integration* that, in its current

[1] http://www.sei.cmu.edu/cmmi/

version, contains three different models: CMMI for Acquisition (CMMI-ACQ) [5], which is used for acquiring products and services; CMMI for Development (CMMI-DEV) [6], for developing products and services; and CMMI for Services (CMMI-SVC) [7], for providing superior services.

The CMMI models focus on the following dimensions that affect organizational improvement: people, procedures and methods, tools and equipment, and processes.

From the number of maturity models that have been defined we present here six, all related to SET-MM and largely inspired by CMM or designed to work in conjunction with it. While the first three models (namely, the Testing Maturity Model, the Measurement Capability Maturity Model and the Software Measurement Process Capability Maturity Model) focus on process maturity, the other three focus on technology maturity (as our model does).

The *Testing Maturity Model* (TMM) [8] was designed to assist software development organizations in evaluating and improving their testing processes.

It covers nine different attributes of a mature testing process: testing policies, test life cycle, test planning process, test group, test process improvement group, test-related metrics, tools and equipment, controlling and tracking mechanism, and product quality control.

The *Measurement Capability Maturity Model* (M-CMM) [9] enables organizations to assess their software and software process measurement capabilities and provides organizations with directions for improving their measurement capability.

It covers the following main areas: measurement focus, measurement design, measure collection, measure analysis, technology support, measurement feedback, measurement training, and measurement management.

The *Software Measurement Process Capability Maturity Model* (SMP-CMM) [10] helps organizations to assess their measurement processes and provides guidelines for improving them.

It supports improvement in five areas: metrics plan, data collection, data gathering, data analysis, and feedback activity.

Daskalantonakis et al. [11] defined a measurement technology maturity model for assessing the software measurement technology of an organization.

Their maturity model covers ten different themes: formalization of the development process, formalization of the measurement process, scope of measurement, implementation support, measurement evolution, measurement support for management control, project improvement, product improvement, process improvement, and predictability.

Wettstein and Kueng [12] defined a maturity model for performance measurement systems, that is, systems that track and manage the performance of an organization (or part of it).

Their maturity model covers six different dimensions: quality of measurement process, scope of measurement, data collection, data storage, use of measures, and communication of results.

Gao et al. [13] define five levels to evaluate the maturity level of software test automation in the test processes of an organization.

In their work, maturity is evaluated through two different perspectives regarding the existence of systematic solutions and tools that support a) the different tasks to be performed during software testing; and b) the measurement of the testing process.

If we classify the main areas covered in these maturity models according to the four improvement dimensions (people, procedures and methods, tools and equipment, and processes) taken into account in CMMI, it can be observed how all these approaches cover the four dimensions and emphasize the areas related to processes while understate those related to people.

While the model presented in this paper is quite similar to the maturity models described above, the main difference is that these maturity models focus on an organization, whereas in our case the focus is on a research field.

This entails two major distinctions between SET-MM and the other models. First, and contrarily to the maturity models presented, we do not cover processes in our model. This is so mainly because at a research field level it is not possible to define, measure, or control any process since the field is composed of highly distributed and heterogeneous members. The second distinction is that data themselves are main topics in SET-MM since the ability to improve largely relies on the capacity of reusing the different evaluation data (e.g., workflows, test data, results). By contrast, in the other maturity models data are secondary issues.

III. Research Methodology

This section presents the scope of the work presented in this paper and the process followed to define the maturity model.

As mentioned above, the maturity model here presented is solely focused on software products and does not cover software processes.

Besides, this maturity model is grounded in the notion of evaluation as defined by the ISO/IEC 14598 standard on software product evaluation [14] and in the following evaluation entities: in any *evaluation* a given set of *tools* are exercised, following a given *evaluation workflow* and using determined *test data*. As an outcome of this process, a set of *evaluation results* is produced.

These entities are depicted in Figure 1. A detailed description of them and of their life cycles can be found in [15].

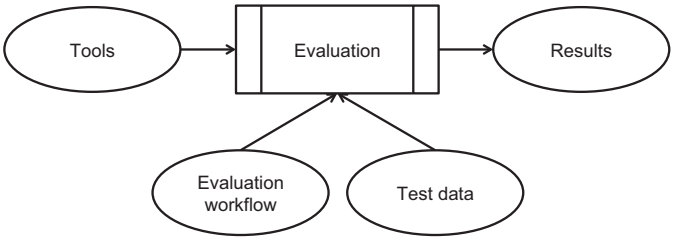

Fig. 1. Main entities in a software evaluation scenario

Furthermore, any evaluation activity is always expected to be the input of some decision-making process. Niessik and

van Vliet [16] exemplify this by proposing a generic process model for measurement-based improvement in organizations; such a model is composed of a measurement cycle followed by an improvement one in which changes in the organization are implemented based on the measurement results.

In our case, the scope is limited to the measurement (i.e., evaluation) cycle; thus, it does not cover improvement, since the analysis and change processes of a research field cannot be controlled nor monitored.

The process followed to define the maturity model contains the same steps than those defined by [11]:

1) First, we identified the set of assumptions upon which the different software evaluation technology maturity levels are defined. Each of these assumptions defines one or more themes (i.e., aspects) that influence maturity.
2) From these themes, five evolutionary stages were defined; these stages have to be followed by any research field in order to reach the highest level of maturity for that particular theme.
3) Then, level *i* of the maturity model corresponds with the *i-th* stage of the themes used to characterize and evaluate the software evaluation technology maturity.

IV. Work Assumptions

This section enumerates the assumptions taken into account while defining the maturity model.

These assumptions (and their corresponding themes) were derived by analysing a) existing maturity models and abstracting their main concepts; and b) evaluations performed in the semantic research field, focusing on the technologies that support these evaluations.

Assumption 1. Software evaluation is facilitated by a well-defined software evaluation workflow and such well-defined workflow will very likely yield quality evaluation results. Furthermore, having both a common framework for defining software evaluations and the means for easily defining evaluation workflows and also automating them is a significant factor that contributes greatly to the improvement of such workflows. Therefore, the following theme is important: *Formalization of the evaluation workflow*.

Assumption 2. Automation of software evaluation tasks enables to perform cost-efficient software evaluations and to diminish manual errors in them. Furthermore, the only way of managing and processing large quantities of software evaluation data is through the use of dedicated evaluation infrastructures. Therefore, the following theme is important: *Software support to the evaluation*.

Assumption 3. The quality of a software evaluation workflow depends on whether such workflow can be applied to different and heterogeneous types of software products. Only by applying the same evaluation workflow across different types of software and under different settings we can validate our hypotheses and conclusions. Therefore, the following theme is important: *Applicability to multiple software types*.

Assumption 4. The automated generation and manipulation of test data for software evaluations helps to focus on how to define test data instead of on how to manage them. Moreover, test data that can be used in different software evaluations are easier to understand, and the results obtained from them are easier to interpret. Therefore, the following theme is important: *Usability of test data*.

Assumption 5. Software evaluation results that are described in some machine-processable format enable the automated integration and exploitation of such results. This integration of results allows the compilation of a significant body of evaluation results, which leads to the exploitation of such results in unexpected ways. Therefore, the following theme is important: *Exploitability of results*.

Assumption 6. The quality of a software evaluation increases when different teams with different viewpoints define and support such evaluation. A software evaluation is more respected when it is supported not by a single team or organization (e.g., the software developers) but by multiple teams (e.g., software providers, users) or by the whole research field. Therefore, the following theme is important: *Representativeness of participants*.

V. Maturity Levels of Software Evaluation Technology

This section describes the five maturity levels of the Software Evaluation Technology Maturity Model. These levels and the goals to be achieved in each of them for a defined theme are presented in Table I.

What follows next is a general description of each level.

A. Level 1. Initial

At this level, a single team defines and carries out the evaluation workflow, which is specific to certain evaluation settings, is informally defined, and is manually performed with no software support. Evaluation is applied to a small number of software products of the same type, using test data not formally defined. The results obtained in the evaluation are also informally described, which makes them impossible to verify.

B. Level 2. Repeatable

At this level, the evaluation workflow is completely defined by one or a few teams and, although it is repeatable, it is still specific of certain evaluation settings. Evaluation software has been developed to partially support the evaluation of a small number of software products of the same type; these software products require to implement evaluation-specific mechanisms so they can be integrated with the evaluation software. The test data used in the evaluation are thoroughly described, and the evaluation results are defined in a machine-processable format, which allows combining the results of the evaluated software products.

C. Level 3. Reusable

At this level, several teams define an evaluation workflow that completely covers one type of software products and that can be reused to evaluate with different test data different

TABLE I
LEVELS AND THEMES OF SOFTWARE EVALUATION TECHNOLOGY MATURITY

Level	Formalization of the evaluation workflow	Software support to the evaluation	Applicability to multiple software types	Usability of test data	Exploitability of results	Representativeness of participants
Initial	Ad-hoc workflow informally defined.	Manual evaluation. No software support.	Small number of software products of the same type.	Informally defined.	Informally defined. Not verifiable.	One team.
Repeatable	Ad-hoc workflow defined.	Ad-hoc evaluation software.	Small number of software products of the same type. Ad-hoc access to software products.	Defined.	Machine-processable. Combined for some software products of the same type.	One or few teams.
Reusable	Technology-specific workflow defined.	Reusable evaluation software: - multiple software products. - multiple test data.	Multiple software products of the same type. Generic access to software products.	Machine-processable.	Machine-processable. Combined for many software products of the same type.	Several teams.
Integrated	Generic workflow defined. Machine-processable and built reusing common parts. Evaluation resources built upon shared principles.	Evaluation infrastructure: - multiple types of software products. - multiple test data.	Multiple software products of different types. Generic access to software products.	Machine-processable. Reused across evaluations.	Machine-processable. Combined for many software products of different types.	Several teams. Stakeholders.
Optimized	Generic workflow defined. Machine-processable and built reusing common parts. Evaluation resources built upon shared principles. Measured and optimized.	Federation of evaluation infrastructures: - autonomous infrastructures. - interchange of evaluation resources. - data access and use policies.	Multiple software products of different types. Generic access to software products. Support any software product requirement.	Machine-processable. Reused across evaluations. Customizable, optimized and curated.	Machine-processable. Combined for many software products of different types. High availability and quality.	Community.

characteristics of such software products. This workflow is supported by evaluation software that can be used to assess any software product of the type covered by the evaluation; the software product must have previously implemented the required mechanisms to be integrated with the evaluation software. Test data and evaluation results are machine-processable; therefore, they can be reused. Furthermore, the results can be combined for all the software products of the same type.

D. Level 4. Integrated

At this level, several teams in collaboration with relevant stakeholders (e.g., users or providers) define a generic evaluation framework that can be used with any type of software product. This generic framework for software evaluation allows building evaluation resources (i.e., evaluation workflow, tools, test data, and results) upon shared principles and reusing common parts. Here, evaluation workflows are defined in a machine-interpretable format so they can be automated. An evaluation infrastructure gives support both to the evaluation of multiple types of software products, taking into account their different characteristics, and to the management of the different evaluation resources. Test data can be reused across different evaluations, and the evaluation results can be combined for software products of different types.

E. Level 5. Optimized

At this level the whole community has adopted a generic framework for software evaluation in which evaluation workflows are measured and optimized. The centralized scenario of the previous levels has now evolved into a federation of autonomous evaluation infrastructures. These evaluation infrastructures must support not only the evaluation workflow but also new requirements, such as the interchange of evaluation resources or the implementation of policies for data access, interchange, and use. This federation of infrastructures permits satisfying any software or hardware requirements of the different software products; customizing, optimizing, and curating test data; and improving the availability and quality of the evaluation results.

One of the notions behind the maturity model, as Figure 2 shows, is that a higher maturity level implies higher integration of evaluation efforts in one field, ranging from isolated evaluations in the lower maturity level to fully-integrated evaluations in the higher level. In this scenario, maturity evolves from a starting point of decentralized efforts into centralized infrastructures and ends with networks of federated infrastructures.

Another notion to consider in this model is that of cost. While the cost of defining new evaluations decreases when the maturity level increases, mainly due to the reuse of existing resources, the cost associated to the evaluation infrastructure (hardware and infrastructure development and maintenance) significantly increases.

VI. ASSESSMENTS IN THE SEMANTIC RESEARCH FIELD

This section presents how we have used SET-MM to assess the maturity of software evaluation technologies in a specific research field.

Other maturity models provide appraisal methods for comparing with the maturity model. However, we do not propose any appraisal method because our scope is a whole research field and, therefore, it would be difficult to obtain objective metrics since any judgment would be subjective.

Therefore, our approach has been, first, to identify some evaluation efforts that stand out because of their impact in the field and, second, to try to assess the maturity of the software evaluation technologies used in them.

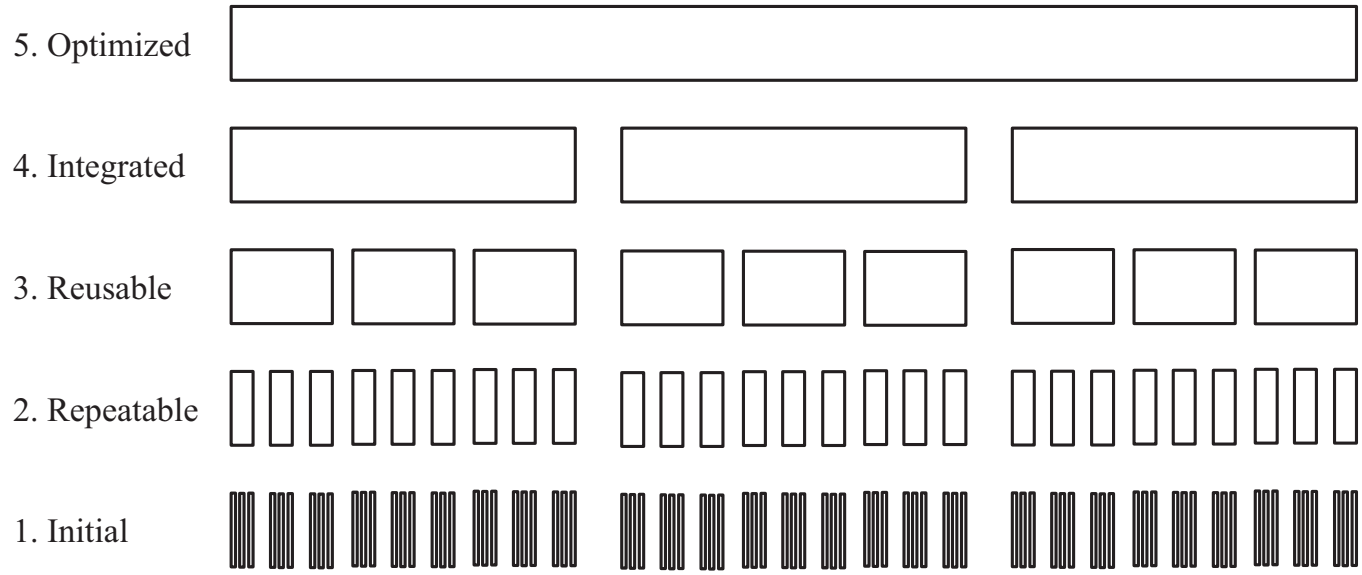

Fig. 2. The SET-MM maturity levels.

A. The Semantic Research Field

The notion of Semantic Web appeared at the beginning of the 21st century [17] and, since then, it has become a research field on its own.

The idea behind the Semantic Web is to have mechanisms to express knowledge and data in the Web so that it can be properly exploited by computers in an automated way.

The way of expressing such knowledge is through ontologies (explicit, structured models of the terminology and conceptual structures used in an application domain), which provide the basis for interpreting and relating data from different sources and that can be used to derive implicit knowledge about data.

One key requirement to make the Semantic Web real is the availability of technologies capable of managing and processing these data at web scale. Because of this, a huge number of research efforts have been devoted to the development and evaluation of semantic technologies.

B. Maturity of Evaluation Efforts

Evaluations in the semantic research field, as in any other field of research, are highly frequent and their main goal is to validate research.

However, if we analyse the technologies used to support those evaluations, we can see that in most of the cases the maturity of such evaluation technologies is at the *Initial* level, which makes almost impossible to reproduce any evaluation.

Nevertheless, there are still some efforts we should highlight because of their maturity in terms of software evaluation technologies.

1) The Lehigh University Benchmark: The Lehigh University Benchmark (LUBM) [18] is the benchmark most used in the semantic field. This benchmark can be used to evaluate the efficiency of ontology storage and reasoning systems and it is composed of a synthetic generator of test data, a set of test queries to be issued to the system, and a test module to automate execution.

LUBM's evaluation technology is at the *Repeatable* level in most of the themes, except in the next two aspects, in which it improves.

The usability of test data is between the *Reusable* and the *Integrated* levels, since the clear definition of test data and their automated generation largely facilitates their reuse across evaluations.

Additionally, the representativeness of participants is at the *Integrated* level because, even if the benchmark was defined by a research group, it is nowadays largely used by researchers and companies.

This high test data maturity can also be observed in the use of LUBM in the field: most of the people that reuse the benchmark only reuse the test data and define their own evaluation settings; furthermore, posterior benchmark improvements have mainly been made on test data [19], [20].

2) The Ontology Alignment Evaluation Initiative: The Ontology Alignment Evaluation Initiative[2] (OAEI) is an international initiative that, since 2004, has been organizing different ontology alignment contests with the goal of establishing a consensus for evaluating ontology alignment methods and their associated tools.

In these contests, ontology alignment systems are compared using a common set of synthetic and real-world tests using a common evaluation framework.

OAEI's evaluation technology is at the *Reusable* level in all themes except one, in which it improves. The representativeness of the participants is at the *Optimized* level; since the OAEI gathers the main stakeholders in the ontology alignment

[2]http://oaei.ontologymatching.org/

topic, its evaluations have become the de facto standard for ontology alignment evaluation.

3) The SEALS Platform: The SEALS Platform [21] is an infrastructure for the evaluation of semantic technologies that offers independent computational and data resources for the evaluation of these technologies.

The SEALS Platform is currently under development in a European project[3] and its goal is to provide the semantic field with an evaluation infrastructure at the *Integrated* level by the end of the project.

To this end, the SEALS Platform provides a common evaluation framework, based on the reusability of evaluation resources, in which different types of semantic technologies can be evaluated. Once the cumulative evaluation results reach a critical mass, the research community will be able to exploit them in novel ways.

VII. Conclusions and Future Work

This paper has presented the SET-MM maturity model that defines the maturity of software evaluation technologies in six different themes.

This maturity model can be used by anyone in a research field willing to assess the maturity of software evaluation technologies, either in the whole field or in specific evaluation initiatives.

SET-MM permits not only to make this assessment but also to guide improvement processes on software evaluation technologies by identifying the different goals to be achieved in each level and for each theme.

This way, the maturity model can be a useful mechanism to increase awareness on the role and on the benefits of software evaluation technologies in research fields.

To illustrate the use of the maturity model, we have assessed the maturity of the software evaluation technologies used in different initiatives within the semantic research field.

Clearly, this assessment is neither exhaustive nor representative of the whole research field. However, it is useful to analyse how software evaluation technologies have been used in different efforts in order to extract those lessons that are worth learning.

CMMI classifies maturity model components into three different categories, namely, required (specific and generic goals), expected (those practices relevant for achieving a goal) and informative (informative material that helps understanding the model). SET-MM currently covers the required components. Future work will deal with the definition of the expected and informative components in order to enrich the model.

Acknowledgment

This work is supported by the SEALS European project (FP7-238975) and by the EspOnt project (CCG10-UPM/TIC-5794) co-funded by the Universidad Politécnica de Madrid and the Comunidad de Madrid. Thanks to Rosario Plaza for reviewing the grammar of this paper.

[3]http://www.seals-project.eu/

References

[1] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute, Tech. Rep. CMU/SEI-93-TR-024, February 1993.

[2] R. Sowden, D. Hinley, and S. Clarke, "Portfolio, Programme and Project Management Maturity Model (P3M3) Version 2.1. Introduction and Guide to P3M3," Office of Government Commerce, United Kingdom, Tech. Rep., 2010.

[3] S. Marshall, "E-Learning Maturity Model Version Two: New Zealand Tertiary Institution E-Learning Capability: Informing and Guiding E-Learning Architectural Change and Development," New Zealand Ministry of Education, Tech. Rep., 26th July 2006.

[4] V. R. Basili, "The role of experimentation in software engineering: past, current, and future," in *Proceedings of the 18th International Conference on Software Engineering (ICSE 1996)*. Berlin, Germany: IEEE Computer Society, 1996, pp. 442–449.

[5] CMMI Product Development Team, "CMMI for Acquisition, Version 1.3," Software Engineering Institute, Tech. Rep. CMU/SEI-2010-TR-032, November 2010.

[6] ——, "CMMI for Development, Version 1.3," Software Engineering Institute, Tech. Rep. CMU/SEI-2010-TR-033, November 2010.

[7] ——, "CMMI for Services, Version 1.3," Software Engineering Institute, Tech. Rep. CMU/SEI-2010-TR-034, November 2010.

[8] I. Burnstein, T. Suwanassart, and R. Carlson, "Developing a testing maturity model for software test process evaluation and improvement," in *Proceedings of the IEEE International Test Conference on Test and Design Validity (ITC 1996)*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 581–589.

[9] F. Niessink and H. van Vliet, "Towards mature measurement programs," in *Proceedings of the 2nd Euromicro Working Conference on Software Maintenance and Reengineering (CSMR 1998)*. Florence, Italy: IEEE Computer Society, March 8–11 1998, pp. 82–88.

[10] M. YongGang and D. JianJie, "Software measurement process capability maturity model," in *Proceedings of the Second International Conference on Computer Modeling and Simulation (ICCMS 2010)*. Sanya, China: IEEE Computer Society, 2010, pp. 400–402.

[11] M. K. Daskalantonakis, R. H. Yacobellis, and V. R. Basili, "A method for assessing software measurement technology," *Quality Engineering*, vol. 3, no. 1, pp. 27–40, 1990-91.

[12] T. Wettstein and P. Kueng, *Management Information Systems 2002 – GIS and Remote Sensing*. Southampton: WIT Press, 2002, ch. A Maturity Model for Performance Measurement Systems, pp. 113–122.

[13] J. Z. Gao, H.-S. J. Tsao, and Y. Wu, *Testing and Quality Assurance for Component-Based Software*. Norwood, MA, USA: Artech House, Inc., 2003.

[14] *ISO/IEC 14598-6: Software product evaluation - Part 6: Documentation of evaluation modules*. ISO/IEC, 2001.

[15] R. García-Castro, M. Esteban-Gutiérrez, M. Kerrigan, and S. Grimm, "An ontology model to support the automatic evaluation of software," in *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010)*. Redwood City, CA, USA: Knowledge Systems Institute, July 1–3 2010, pp. 129–134.

[16] F. Niessink and H. van Vliet, *A Pastry Cook's View on Software Measurement*. Wiesbaden, Germany: Deutscher Universitaetsverlag, 1998, pp. 109–126.

[17] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[18] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A Benchmark for OWL Knowledge Base Systems," *Journal of Web Semantics*, vol. 3, no. 2, pp. 158–182, 2005.

[19] L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, and S. Liu, "Towards a complete OWL ontology benchmark," in *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, ser. LNCS, vol. 4011. Budva, Montenegro: Springer, June 11-14 2006, pp. 125–139.

[20] T. Weithöner, T. Liebig, M. Luther, S. Böhm, F. von Henke, and O. Noppens, "Real-world reasoning with OWL," in *Proceedings of the 4th European Semantic Web Conference (ESWC2007)*, ser. LNCS, vol. 4519. Springer, 2007, pp. 296–310.

[21] R. García-Castro, M. Esteban-Gutiérrez, and A. Gómez-Pérez, "Towards an infrastructure for the evaluation of semantic technologies," in *Proceedings of the eChallenges 2010 Conference*, P. Cunningham and M. Cunningham, Eds., Warsaw, Poland, October 27-29 2010.